Lab 06: Transaction Processing Program using Random Access Files

In this assignment you are going to implement file handling functions.

***Note:* Students must demonstrate their lab in the following week of December 7th, 2020 to get the grade. This lab is worth 20marks. A Missed demonstration will result in 25% grade deduction. Late submission and late demonstration to the lab is welcomed but with a loss of marks. That is, you do not earn full marks for late submission. 10% penalty for every day late up to 50%. Work will not be accepted after 10 days. You must talk to your lab professor in case of any unprecedented situation which may result in late submission.**

You are required to submit the properly commented (as per assignment submission standards doc on BrightSpace) text file of your code along with a screenshot of the output.

Program Statement:

The program maintains a bank's account statistics. The program updates existing accounts, adds new accounts, deletes accounts and stores a listing of all the current accounts in a text file for printing. You must assume that the below program (creation of random access file sequentially) has been executed to create the file credit.dat.

```
/*Consider the following statement for the creation of a random-access file sequentially:--
  Creating a credit processing system capable of storing upto 100 fixed-length records. Each
record consists of an account number that will be used as the 'record key' (first name, last
name and a balance). The resulting program must be able update an account, insert a new
account record, delete an account and list all the account records in a formatted text file for
printing. Make use of a random-access file*/

#include<stdio.h>
struct clientInfo
{
	int AccNo;
	char lastName[20];
	char firstName[20];
	double accBalance;
};
int main(void)
{
	int i;			/*counter used to count from 1-100*/
	struct clientInfo randomClient = {0, "", "", 0.0};		/*create clientInfo with
default information*/
	 FILE *fptr;
```

```
        if((fptr=fopen("credit.dat", "wb"))==NULL)
        {
                printf("File could not be opened");
        }
        Else
        {
                for(i=1;i<=100;i++)           /*output 100 random blank records to file*/
                {
                        fwrite(&randomClient, sizeof(struct clientInfo), 1, fptr);
                }
        fclose(fptr);
        }
    return 0;
}
```

The program has five options:

1) Option 1 calls function textFile to store a formatted list of all the accounts in a text file called accounts.txt that may be printed later. To implement this, the function uses fread and the sequential file access techniques. After choosing option 1, the file accounts.txt contains:

```
accNo.          lastName            firstName           accBalance
29              Brown               Peter               -121.01
33              Dunn                George              142.23
37              Barker              Gee                 0.01
88              Smith               David               248.15
89      Stone               Sam                 35.68
```

2) Option 2 calls the function updateRecord to update an account. The function will only update a record that already exists, so the function first checks to see if the record specified by the user is empty. The record is read into structure client with fread, then member acctNo is compared to 0. If it's 0, the record contains no information, and a message is printed stating that "the record is empty". Then, the menu choices are displayed. If the record contains information, function updateRecord inputs the transaction amount, calculates the new balance and rewrites the record to the file. A typical output for option 2 is:---

```
Enter account to update (1-100):    37
37      Barker  Gee     0.01
Enter charge (+) or payment (-): +81.95
37 Barker       Gee     81.96
```

3) Option 3 calls the function newRecord to add a new account to the file. If the user enters an account number for an existing account, newRecord displays an error message that the record already contains information, and the menu choices are printed again. A typical output for option 3 is:--

```
Enter new account number (1-100): 21
Enter lastName, firstName, balance
Johnson Sarah 247.845
```

4) Option 4 calls function deleteRecord to delete a record from the file. Deletion is accomplished by asking the user for the account number and reinitializing the record. If the account contains no information, deleteRecord displays an error message that the account does not exist.
5) Option 5 terminates program execution.